d-BAME: Distributed Blockchain-based Anonymous Mobile Electronic Voting

Ehab Zaghloul, Tongtong Li and Jian Ren

***Abstract*—Electronic voting (e-voting) presents a convenient and cost-effective alternative to current paper ballot-based voting. It provides many benefits such as increased voter turnout and accuracy in the decision-making process. While presenting many improvements, e-voting still faces serious security challenges that hinder its adoption, especially when designed to be run over mobile devices. In this paper, we propose a novel remote e-voting model for large-scale elections by proposing the participation of two conflicting parties to ensure election integrity and accountability. Our scheme can be implemented in IoT devices such as smartphones, which we believe can significantly increase voter turnout of the election process. Our proposed work is secure and preserves voter privacy through secure multi-party computations performed by parties of differing allegiances. It also leverages a blockchain running smart contracts as a publicly accessible and tamper-resistant bulletin board to permanently store votes and prevent double-voting. In our security and privacy analysis, we show that our proposed scheme is secure against potential security threats and provides voter anonymity. We show orthogonality between universal verifiability and coercion-resistance in our proposed scheme, allowing an election to favor one over the other. Our performance analysis and smartphone simulation results show that the proposed scheme is practical for large-scale elections.**

***Index Terms*—Remote e-voting, voter anonymity, universal verifiability, coercion-resistant, unlinkability, blockchain**

I. Introduction

More than half the world's countries are classified as democratic nations, employing governments that enforce and secure their democracies. While these governments may vary in structure, they all grant eligible members the ability to exercise their power by voting. However, guaranteeing that a democratic election is *free* and *fair* still remains a challenge for most governments. Let alone, proving the freeness and fairness of the election to everyone, especially to the losing candidates, is an even bigger challenge.

A free election should entail multiple imperative features [1]. Before voting, proper voter registration is required to grant voting rights only to eligible voters. Voters should be able to remain anonymous, maintaining an election free of ballots that could be linked to their voters. Furthermore, to ensure that votes are tallied properly, verifiability should also be integrated to prove to everyone the legitimacy of the election results and avoid controversy. Concurrently, for an election to be fair, all eligible voters should have equal registration and ballot casting availability and accessibility regardless of any limitations such as geographical location or economic status. This means that voters that are unable to physically access their poll-sites, for example, absent personnel serving in the military, should be able to cast their ballots remotely while maintaining the equivalent requirements of a free election. A fair election should also maintain the secrecy of the cast ballots throughout the voting phase to prevent last-minutes voters from skewing the final count.

Nowadays, the majority of democratic elections are run or operated using in-person isolated poll-sites with rigorous monitoring in an effort to uphold a free and fair election. This requires voters to physically cast their ballots at predetermined public sites. The most widely utilized casting techniques include ballot box elections where voters insert their paper ballots into a box, scan them using optical scanners, or vote using a Direct-Recording Electronic (DRE) voting machine. While ballot box elections may cultivate many of the aforementioned features, they fall short in verifiability and require significant trust in the election organizers and tallying authority to behave honestly. While incorporating computerized systems such as optical scanners and DREs along with cryptographic primitives could help reduce the human factor intervention and may even offer verifiability, these systems still present computer vulnerabilities as proven by various research [2], [3]. Various schemes such as [4], [5] aim to minimize such vulnerabilities and provided end-to-end (E2E) verifiable voting. However, the mandatory requirement of voting at a poll-site using either technique interferes with the availability and accessibility fairness requirements and overall voter turnout. To overcome this issue, some elections may permit remote voting through absentee/mail-in ballots allowing voters to vote at other poll-sites or return their ballots via mail. Yet, these method raise concerns that ballots may be subject to fraud/manipulation during the process of transmission.

Recent research has presented several remote voting systems [6]–[12] that rely heavily on cryptography and aim to achieve the major desired requirements. Unfortunately, such systems have not been adopted by large-scale elections since they cannot achieve the same level of freeness and fairness achieved by conventional poll-site voting or they are only feasible and applicable to small-scale elections. In this paper, we introduce a novel voting scheme that enables free and fair large-scale elections. Our proposed scheme leverages the existence of at least two parties of an election with different allegiances that engage in a multi-party computation along with the voters. We assume that it conflicts with the interest of these parties to collude or exchange any information during

This work was supported in part by NSF under Grant CCF-1919154 and Grant ECCS-1923409. (Corresponding author: Jian Ren.)

The authors are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824-1226, Email: {ebz, tongli, renjian}@msu.edu

the election process that may sacrifice the winning chances of the candidates they support. All computations can be performed remotely at the convenience of the parties involved. In addition, voters are able to cast their votes from a mobile device and verify whether their votes have been cast and counted properly. Voter verifiability is based on randomly generated values that even if shared with coercers willingly, will not provide any information on how the voters have voted. Furthermore, we utilize a blockchain that acts as a publicly accessible bulletin board that voters cast and store their votes to. No computations of our proposed scheme are performed over the blockchain which will circumvent the scalability or cost issues of blockchain in large-scale elections.

The model presented in this paper is primarily based on the US general election. It expands significantly on [13] in that we show orthogonality between universal verifiability and coercion-resistance in our proposed scheme, allowing an election to favor one over the other. We also prove that it is computationally infeasible to link any vote to the voter or multiple votes cast by one voter. The main contributions presented in this paper can be summarized as follows:

- We develop d-BAME, a novel distributed anonymous mobile electronic voting scheme for large-scale elections. Our proposed scheme is designed to provide several design trade-offs for elections with different requirements. It builds over secure multi-party computations, allowing voters to cast their votes remotely using mobile devices such as smartphones, storing them over a blockchain permanently and irreversibly.
- We conduct comprehensive security and privacy analyses of d-BAME. We provide formal proofs to prove that d-BAME is secure against double voting, preserves voter anonymity, provides coercion-resistance and ballot unlinkability, and prevents manipulation to the election results.
- We present a performance analysis for d-BAME and compare it to two existing blockchain-based schemes [14], [15]. We calculate the different number of operations performed by each involved party in the entire voting process for all schemes. Our analysis shows that d-MABE outperforms both schemes.
- We implement a desktop application and an iOS mobile application for d-BAME and the corresponding smart contracts which we deploy over the Ropsten Ethereum testnet. We follow with an empirical evaluation showing the feasibility of d-BAME to be run on a mobile device. Our results show that, even when running it under encryption key sizes of 4096 bits, voters can cast their votes via mobile devices in less than a minute.

The rest of this paper is organized as follows. In Section II, the related work is reviewed. In Section III, preliminaries are introduced that summarize key concepts used in this research. Next, in Section IV, the problem formulation is described, outlining the system model and our design goals. In Section V, our proposed scheme is presented in detail outlining the proposed algorithms. Following that, in Section VI we formally prove the security and privacy of our proposed scheme. In Section VII, performance analysis and evaluation of d-BAME are conducted and give our empirical results in Section VIII. Following that, we discuss design trade-offs in Section IX. Finally, in Section X, a conclusion is drawn to summarize the work done in this research.

II. Related Work

Previous work that achieves coercion-resistance and remote e-voting date back to 2005 when the work by Juels *et al.* [6] was introduced and later refined resulting in Civitas [7]. Although proven to be secure, the security came at the price of tabulation which is quadratic in respect to the total number of ballots being submitted in an election. Shortly, Helios [8] was proposed as a web-based open-audit voting system for elections where coercion is not a serious problem. The system achieves privacy using mixnets [9] and was later improved by Demirel *et al.* [10] by replacing the mixnets with homomorphic encryption and multi-party tallying. However, these systems primarily focus on universal verifiability while intentionally not taking coercion-resistance into account.

In contrast, Selections [11] is a system that was proposed in which voter authentication relies on possession of certain voter passwords, allowing them to generate panic passwords in cases of coercion. This system relies on zero-knowledge proofs during the vote casting phase. Other systems such as [12], use a linear-time scheme to remove duplicated votes which may be submitted by voters to avoid coercion. This system also relies on voters indicating which electoral roll their votes belong to so that tallying authorities can identify which votes should be included in the total count. This results in faster tallying during the tabulation process. However, it requires additional trust in the elected trustees to add dummy ballots to make the system coercion-resistant.

Based on concepts from [11] and [12], a protocol was introduced in [16] that requires voters to specify an anonymity set where each voter claims to be one of the voters within the set. This resulted in additional voter overhead costs during the authentication phase. Later, Zeus [17] was proposed following the initial framework of Helios [8] where mixing is performed using external agents. Although it provides universal verifiability, the system is not coercion-resistant as it provides voters with receipts at the end of voting.

DEMOS-2 [18] aims to be E2E verifiable through a voter supporting device (VSD) and trustees. The election authority uses voters‘ coins to generate and prove the security of a common reference string (CRS). The CRS can be used by the voters to affix non-interactive zero-knowledge (NIZK) proofs to their ciphertexts and by election authority to prove via a NIZK the tally correctness at the end of the election. Later, BeleniosRF [19] was introduced to also provide stronger receipt-freeness using signatures on randomizable ciphertexts (SRC) to prevent the voters from proving how they voted.

More recently, Blockchain-based voting schemes began to appear, taking advantage of its irreversible, permanent, and smart contract features. Smart contracts for boardroom small-scale voting systems [14] were implemented over the Ethereum blockchain using the Open Vote Network [20].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3074877, IEEE Internet of Things Journal

The main advantages of this system are that it is completely decentralized, provides self-tallying, and achieves E2E characteristics. However, the system utilizes zero-knowledge proofs which increases its overall computational costs. The majority of these cryptographic computations are implemented in smart contracts and executed over the blockchain, limiting the protocol to small-scale elections. In addition, the system is not coercion-resistant and can be manipulated by last-minute voters since they can tally the results before casting their votes. To circumvent this issue, the scheme implements an optional additional round where voters initially cast a hashed version of their votes as a commitment before casting their actual encrypted votes. In this case, although last-minute voters can still compute the election results before casting their votes, they can no longer change their selection to manipulate the results. However, this method requires additional smart contract callings and computations, imposing additional costs.

Another small-scale election was also introduced in [21] that relies on expensive homomorphic encryption primitives to preserve voter privacy. Similarly, incorporating such resource-intensive cryptographic computations requires significant costs and limits the scheme to small-scale elections.

Currently, there are no widely acceptable large-scale blockchain-based voting schemes developed in literature. Although [15] has been proposed utilizing cryptographic techniques like Paillier encryption, proof-of-knowledge, and linkable ring signature, deploying such a scheme is computationally inefficient and expensive. In addition to the informal security analysis provided in this work, voters have to interact with a smart contract deployed over the blockchain to cast and verify their votes. This process requires the smart contract to perform expensive verification operations during both vote casting and tabulation phases, which may have serious performance issues, and security and privacy concerns.

III. Preliminaries

A. Blockchain

Blockchains are immutable and irreversible Distributed Ledger Technologies (DLTs) that allow data to be stored globally across all nodes of a peer-to-peer (P2P) network while maintaining data integrity. They are generally categorized as public (permissionless) or private (permissioned), depending on the node capabilities to participate in governing the blockchain. Permissionless blockchains allow any node connected to the P2P network to participate in the consensus protocol while permissioned blockchains allow only a predefined set of nodes. In this work, we utilize a permissionless blockchain as a bulletin board where voters cast their votes by interacting with the election smart contracts deployed over the blockchain. Thus, we briefly discuss the main components of blockchains and refer readers to [22] for further reading.

B. Decisional Diffie-Hellman (DDH) Assumption

The DDH assumption is a computational hardness assumption and is defined as follows:

Let $\mathbb{G}$ be a group of prime order p, g be a generator, and $a, b, c \in_r \mathbb{Z}_p^*$, where $\in_r$ denotes chosen at random.

It is infeasible for the adversary to distinguish between any given (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), i.e., an algorithm $\mathcal{A}$ that outputs a guess $c = ab$, has advantage ε in solving the DDH problem in $\mathbb{G}$ if:

$$\left|\mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^{ab})=1] - \mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^c)=1]\right| > \varepsilon, \quad (1)$$

where the value 1 denotes *true*. The DDH assumption holds if no probabilistic polynomial time (PPT) algorithm has a non-negligible advantage in solving the DDH problem.

IV. Problem Formulation

Large-scale elections typically involve at least two parties with conflicting allegiances competing to win an election. The challenge is to provide a complete voting process that all voters and running candidates can trust. It would be ideal to allow eligible voters to cast their votes remotely from anywhere while securing the integrity of the election and the safety of the voters.

A. System Components

The general model of our proposed scheme consists of a minimum of six entities, each with a distinct role.

Voters: The eligible set of voters $\{v_i \in \mathcal{V} \mid 1 \leq i \leq n\}$ that are granted the right to cast a vote in an election. This set is public and subject to audits to prove that only eligible voters can vote.

Registrar $\mathcal{R}$: The first election organizing entity, responsible for generating unique and random digital ballots to be shared with voters anonymously. It cannot link a digital ballot to its assigned voter.

Moderator $\mathcal{M}$: The second election organizing entity, responsible for concealing the identities of voters and delivering the ballots to them anonymously. It cannot reveal the concealed digital ballots as it delivers them to the voters, hence, it cannot link a digital ballot to its assigned voter.

Election candidates: The eligible set of candidates $\{\mathsf{cand}_k \in \mathcal{C} \mid 1 \leq k \leq c\}$ running in an election.

Blockchain network: A non-trusted peer-to-peer network that maintains a publicly accessible blockchain and runs the election smart contract. The network nodes cannot link cast votes to voters or differentiate between valid and invalid votes.

Tallying authority: A party that performs vote tabulation at the end of the casting phase. This task is performed and monitored publicly and therefore does not require any trust. It can be performed by anyone monitoring the blockchain.

B. Security Model and Design Goals

Our proposed scheme aims at satisfying the following design goals:

Distributed trust: We assume that an election will be organized by a registrar $\mathcal{R}$ and a moderator $\mathcal{M}$ with conflicting interests. Neither $\mathcal{R}$ nor $\mathcal{M}$ is fully trusted by all voters. However, due to the conflicting interests between them, they are unlikely to collude.

Voter eligibility: Voting is limited to eligible voters. This requires proper voter registration that determines and confirms the eligibility of voters, granting them voting rights.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3074877, IEEE Internet of Things Journal

TABLE I: Notations Summary

Symbol	Definition
v_i	i^{th} voter, where $\{v_i \in \mathcal{V} \mid 1 \leq i \leq n\}$
$\mathcal{R}$	registrar
$\mathcal{M}$	moderator
cand_k	candidate, where $\{\mathsf{cand}_k \in \mathcal{C} \mid 1 \leq k \leq m\}$
x_i, y_i	v_i's private and public keys, respectively
x_r, y_r	$\mathcal{R}$'s private and public keys, respectively
x_m, y_m	$\mathcal{M}$'s private and public keys, respectively
bal_i	i^{th} ballot, where $\mathcal{B} = \{\mathsf{bal}_i = \mathsf{EG\text{-}Sign}(t_i) \mid i \in \mathbb{Z}_n\}$
b_i	i^{th} blind factor, used to conceal v_i's public key y_i
k_i	i^{th} one-time key used to encrypt bal_i
ebal_i	i^{th} encrypted ballot
Q_i	i^{th} ephemeral key
eb_i	encrypted b_i
T	v_i's ballot and vote, where $T = (\mathsf{bal}_i \parallel \text{Vote})$
B_i	v_i's double encrypted T

Double-voting resistant: Each eligible voter is entitled to only a single vote counted toward the election. While submitting multiple votes is permitted, only one vote will be counted toward the tallied results as predefined by the election.

Anonymity: A cast vote cannot be linked to the identity of a voter. This protects voters, allowing them to freely voice their desired opinions.

Coercion-resistant: Remote voting may expose voters to coercion. If subject to coercion, our goal is to provide voters the capability to cast votes as instructed under coercion, whilst still protecting their actual votes.

Voter verifiability: Voters can verify that their votes have been cast properly and counted toward the election results.

Universal verifiability: Anyone can verify that all legitimate votes have been counted correctly.

Election result manipulation-resistant: Last-minute voters cannot manipulate the election results in a close race. This requires concealing all votes until the voting phase is over.

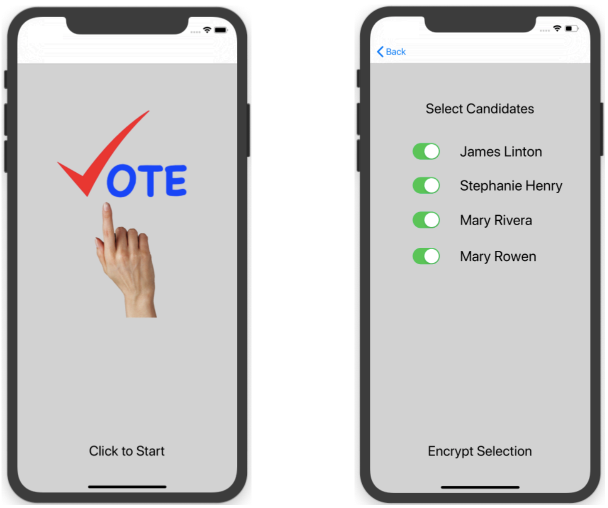
Network adversary: We assume adversaries are capable of monitoring public communication channels and performing general network-level attacks. By design, our proposed scheme builds on cryptographic operations limiting the impact of network adversaries to at most DoS attacks since all operations, such as ballot acquiring and vote casting, can all be verified by the voters.

V. The Proposed Scheme

In this section, we present the sequential phases of our proposed d-BAME scheme, each occurring during a specific time frame predefined by the election organizers. Let $\mathbb{G}$ be a publicly chosen multiplicative cyclic group of prime order p and g is a generator of $\mathbb{G}$. Table I, presents a summary of notations used throughout the paper.

A. Setup

The registrar and moderator each generate a key pair. They select a secret key $x_r \in_r \mathbb{Z}_p^*$ and $x_m \in_r \mathbb{Z}_p^*$ and then compute their corresponding public keys as $y_r = g^{x_r} \pmod{p}$ and $y_m = g^{x_m} \pmod{p}$, respectively.



Select Candidates

● James Linton
● Stephanie Henry
● Mary Rivera
● Mary Rowen

(a) application start

(b) select/encrypt selected candidates

Fig. 1: Steps to select the desired candidates

Protocol 1 Voter Registration

```
Input: Public key $y_i$ of $v_i$; hash function $h$.
Output: Digital Signature $\mathsf{EG\text{-}Sign}_{y_r}(y_i)$.
Setup: (Registrar $\mathcal{R}$)
  1: Randomly select secret key $x_r \in_r \mathbb{Z}_p^*$.
  2: Compute public key as $y_r = g^{x_r} \pmod{p}$.
Registration Request: (Voter $v_i$)
  1: Randomly select secret key $x_i \in_r \mathbb{Z}_p^*$.
  2: Compute public key as $y_i = g^{x_i} \pmod{p}$.
  3: Send $y_i$ to $\mathcal{R}$.
Authentication: (Registrar $\mathcal{R}$) If $v_i$ is eligible:
  1: Randomly select $u_i$ with $1 < u_i < p-1$ and $\gcd(u_i, p-1) = 1$.
  2: Compute $w_i = g^{u_i} \pmod{p}$.
  3: Compute $s_i = (h(y_i) - x_r w_i)u_i^{-1} \pmod{p}$.
  4: Send $(w_i, s_i)$ to $v_i$.
```

B. Voter Registration

At this phase, voters are required to prove their voting eligibility to the registrar by providing evidence such as their identities. After validation, voters are added to the electoral roll. Protocol 1 summarizes this process.

1) Voter Key Generation and Registration: Each voter v_i selects a secret key $x_i \in_r \mathbb{Z}_p^*$ and then computes the corresponding public key as $y_i = g^{x_i} \pmod{p}$. Public keys are shared with the registrar during registration.

2) Signing Voter's Public Key: The registrar verifies the eligibility of the voters and signs their public keys. It selects u_i randomly where $1 < u_i < p-1$ and $\gcd(u_i, p-1) = 1$ then computes the following:

$$w_i = g^{u_i} \bmod p, \;\; s_i = (h(y_i) - x_r w_i)u_i^{-1} \bmod p, \tag{2}$$

where (w_i, s_i) is the signature. The registrar adds $y_i \parallel (w_i, s_i)$ to the electoral list which it discloses once the registration phase is complete.

C. Acquiring a Ballot

To cast a vote, each voter must first acquire a digital ballot. Protocol 2 summarizes this process.

1) Ballots Generation: The registrar generates n unique digital ballots $\mathcal{T} = \{t_i \,|\, i \in \mathbb{Z}_n\}$, such that

$$t_i = \mathsf{Election_{ID}} \,\|\, \mathsf{Election_{TimeStamp}} \,\|\, \mathsf{Rand_i},$$

where $\mathsf{Rand}_i \in_r \mathbb{Z}_e$ for some integer e that can ensure that ballots are unique and not easily forgeable. The registrar then digitally signs t_i using ElGamal signature scheme. We denote the set of digitally signed ballots as $\mathcal{B} = \{\mathsf{bal}_i = t_i \,\|\, \mathsf{EG\text{-}Sign}(t_i) \,|\, i \in \mathbb{Z}_n\}$. Next, it performs a permutation π on $\mathcal{B}$ such that:

$$\pi \colon \mathcal{B} \rightarrow \mathcal{B}.$$

2) Voter Permutation: The moderator is an intermediary that conceals voter identities from the registrar during ballot distribution. It generates a one time permuted set σ of $\{1, 2, \ldots, n\}$ such that:

$$\sigma \colon \mathbb{Z}_n \rightarrow \mathbb{Z}_n.$$

3) Requesting a Ballot: Voters initiate the request by sharing their public keys y_i and corresponding signature $\mathsf{EG\text{-}Sign}_{y_r}(y_i) = (w_i, s_i)$ with the moderator.

4) Voter Identity Obscuring: The moderator checks that y_i exists in the published electoral roll list and validates the shared signature (w_i, s_i) corresponding to y_i through the following equation:

$$g^{h(y_i)} \equiv y_i^{w_i} \cdot w_i^{s_i} \pmod p. \tag{3}$$

The moderator can verify that the voter indeed possesses the private key associated with the public key in the electoral roll through a challenge-response based authentication. The moderator can also check other security credentials, such as DOB, SSN, driver's license number, etc., to ensure proper voter authentication. In this event, the moderator obscures y_i by selecting a blind factor $b_i \in_r \mathbb{Z}_p^*$ and computing the following:

$$y_i' = y_i^{b_i} = g^{x_i b_i} \pmod p. \tag{4}$$

The moderator then sends y_i' and $\sigma(i)$ to the registrar.

5) Ballot Assignment and Encryption: Ballots are assigned randomly by the registrar to the blinded voters as follows:

$$\mathsf{bal}_i = \pi(\sigma(i)). \tag{5}$$

To conceal bal_i, the registrar encrypts it under an encryption key k_i derived as:

$$k_i = (y_i')^{q_i} = g^{x_i b_i q_i} \pmod p, \tag{6}$$

where $q_i \in_r \mathbb{Z}_p^*$. Using k_i, the registrar encrypts the ballot as the following:

$$\mathsf{ebal}_i = \mathsf{AES\text{-}Enc}_{k_i}(\mathsf{bal}_i), \tag{7}$$

where $\mathsf{AES\text{-}Enc}$ is the AES encryption function. The purpose of this encryption is to conceal the ballot from the moderator. It enables the registrar to share this ballot with the voter anonymously. For this purpose, it generates an ephemeral key Q_i that would allow the voter to regenerate k_i such that:

$$Q_i = g^{q_i} \pmod p. \tag{8}$$

Finally, the registrar sends ebal_i and Q_i to the moderator.

6) Encrypted Ballot Transmission: Once received, the moderator sends ebal_i and Q_i to the voter along with the encryption of the blind factor b_i as:

$$eb_i = (g^{r_m},\, b_i \cdot y_i^{r_m}) = (c_{i,1},\, c_{i,2}), \tag{9}$$

where $r_m \in_r \mathbb{Z}_p^*$.

7) Deriving Ballot: The voter decrypts eb_i and recovers b_i as:

$$b_i = c_{i,2} \cdot c_{i,1}^{-x_i} = b_i \cdot y_i^{r_m} \cdot (g^{r_m})^{-x_i}. \tag{10}$$

Next, the voter regenerates key k_i as:

$$k_i = (Q_i)^{x_i b_i} = g^{q_i x_i b_i} \pmod p. \tag{11}$$

Finally, the voter decrypts ebal_i as:

$$\mathsf{bal}_i = \mathsf{AES\text{-}Dec}_{k_i}(\mathsf{ebal}_i), \tag{12}$$

where $\mathsf{AES\text{-}Dec}$ is the AES decryption function.

D. Casting Votes

Before casting a vote, voters select the desired candidates they wish to vote for. Fig. 1 represents our designed mobile application showing candidate selection. Next, the voter proceeds with encrypting and casting the vote.

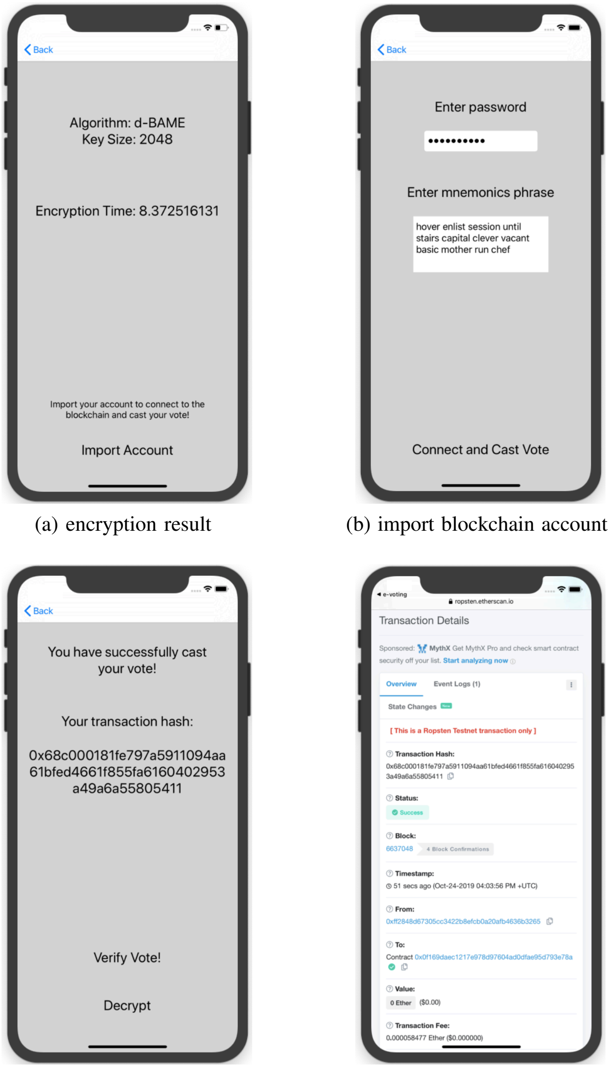
1) Ballot Double-Encryption: The ballot associated with a vote, denoted as B_i, is encrypted under the public keys y_r and y_m of both the registrar and moderator as:

$$B_i = (g^v,\, T_i \cdot (y_r \cdot y_m)^v) = (c_{i,3},\, c_{i,4}), \tag{13}$$

where $v \in_r \mathbb{Z}_p^*$, $T_i = \mathsf{bal}_i \,\|\, \mathsf{vote}_i$, and $\mathsf{vote}_i = (\mathsf{cand}_{i,1}, \mathsf{cand}_{i,2}, \ldots, \mathsf{cand}_{i,m})$ is a sequence of bits representing each candidate such that:

$$\mathsf{cand}_k = \begin{cases} 1, & \text{if voting for } \mathsf{cand}_k, \\ 0, & \text{if voting against } \mathsf{cand}_k. \end{cases} \tag{14}$$

2) Submit Vote: To submit the encrypted ballot, the voter calls the election vote smart contract $\mathsf{SC_{vote}}$ that takes B_i as input. A vote is permanently cast once the result of the smart contract is appended to the blockchain. Fig. 2 is a summary of the steps performed by the voter to cast the encrypted vote. This process involves importing the blockchain account of the voter that allows her to call the $\mathsf{SC_{vote}}$ deployed over the blockchain and integrate the encrypted ballot as input. Once the $\mathsf{SC_{vote}}$ transaction is appended to the blockchain, the vote is cast and the voter receives a confirmation of the vote along with the transaction hash reference. The transaction hash is used to verify that the cast vote is stored permanently over the blockchain.



Back

Algorithm: d-BAME
Key Size: 2048

Encryption Time: 8.372516131

Import your account to connect to the blockchain and cast your vote!

Import Account

(a) encryption result

Back

Enter password

••••••••••

Enter mnemonics phrase

hover enlist session until stairs capital clever vacant basic mother run chef

Connect and Cast Vote

(b) import blockchain account

Back

You have successfully cast your vote!

Your transaction hash:

0x68c000181fe797a5911094aa61bfed4661f855fa6160402953a49a6a55805411

Verify Vote!

Decrypt

(c) block containing cast vote

◂ e-voting

ropsten.etherscan.io

Transaction Details

Sponsored: MythX Get MythX Pro and check smart contract security off your list. Start analyzing now

Overview | Event Logs (1)

State Changes New

[This is a Ropsten Testnet transaction only]

Transaction Hash:
0x68c000181fe797a5911094aa61bfed4661f855fa6160402953a49a6a55805411

Status:
Success

Block:
6637048 4 Block Confirmations

Timestamp:
51 secs ago (Oct-24-2019 04:03:56 PM +UTC)

From:
0xff2848d67305cc3422b8efcb0a20afb4636b3265

To:
Contract 0x0f169daec1217e978d97604ad0dfae95d793e78a

Value:
0 Ether ($0.00)

Transaction Fee:
0.000058477 Ether ($0.000000)

(d) verify vote on blockchain

Fig. 2: Steps to cast and verify vote

3) Publishing Requested Ballots: After the casting votes phase is complete, the registrar publishes the set of requested ballots $\mathcal{B}' \subset \mathcal{B}$ to the blockchain bulletin board, discarding the ballots that have not been requested by registered voters. This prevents the registrar from using any ballots that have not been requested or even creating additional ballots that can be used to cast unregistered votes and sway an election in a specific direction. This allows the moderator to audit the correct number of ballots distributed, while provides ballot universal verifiability since voters can verify that their received ballots are legitimate.

E. Tabulation

Votes that appear on the blockchain within the casting votes phase are collected to be validated and counted. Both the moderator and registrar publicly disclose their secret keys x_m and x_r. The tallying authority decrypts the attached ballots appearing on the blockchain as:

$$T_i = c_{i,4}\cdot c_{i,3}^{-x_m}\cdot c_{i,3}^{-x_r} = T_i\cdot(y_r\cdot y_m)^v\cdot(g^v)^{-x_m}\cdot(g^v)^{-x_r}, \quad (15)$$

Protocol 2 Ballot Acquisition

```
Input: Public key $y_i$ and it's signature $(w_i, s_i)$ of $v_i$.
Output: Ballot $\mathsf{bal}_i$
Ballot Generation (Registrar $\mathcal{R}$):
  1: Generate random ballots as $\mathcal{T} = \{t_i \mid i \in \mathbb{Z}_n\}$.
  2: Digitally sign ballots as $\mathcal{B} = \{\mathsf{bal}_i = t_i \,\|\, \mathsf{EG\text{-}Sign}(t_i) \mid i \in \mathbb{Z}_n\}$.
  3: Perform permutation $\pi\colon \mathcal{B} \to \mathcal{B}$.
Voter Permutation (Moderator $\mathcal{M}$):
  1: Generate permutation as $\sigma\colon \mathbb{Z}_n \to \mathbb{Z}_n$.
Request Ballot (Voter $v_i$):
  1: Send public key $y_i$ and $\mathsf{EG\text{-}Sign}_{y_r}(y_i)$ to the moderator.
Voter Key Validation (Moderator $\mathcal{M}$):
  1: Check $0 < w_i < p$ and $0 < s_i < p-1$.
  2: Verifies $g^{h(y_i)} \equiv y_i^{w_i}, w_i^{s_i} \pmod p$. If Valid, $\Gamma = $ true; otherwise $\Gamma = $ false.
If $\Gamma = $ true:
Voter Key Obscuring (Moderator $\mathcal{M}$):
  1: Randomly select $b_i \in_r \mathbb{Z}_p^*$.
  2: Compute $y_i' = y_i^{b_i} = g^{x_i b_i} \pmod p$.
  3: Send $y_i'$ to $\mathcal{R}$.
Ballot Assignment and Encryption (Registrar $\mathcal{R}$):
  1: Select ballot as $\mathsf{bal}_i = \pi(\sigma(i))$.
  2: Compute key as $k_i = (y_i')^{q_i} = g^{x_i b_i q_i} \pmod p$.
  3: Encrypt ballot as $\mathsf{ebal}_i = \mathsf{AES\text{-}Enc}_{k_i}(\mathsf{bal}_i)$.
  4: Compute ephemeral key as $Q_i = g^{q_i} \pmod p$.
  5: Send $\mathsf{ebal}_i$ and $Q_i$ to $\mathcal{M}$.
Ballot Transmission (Moderator $\mathcal{M}$):
  1: Encrypt $eb_i = (g^{r_m}, b_i \cdot y_i^{r_m}) = (c_{i,1}, c_{i,2})$
  2: Send $\mathsf{ebal}_i$, $Q_i$, and $eb_i$ to $v_i$.
Derive Ballot (Voter $v_i$):
  1: Decrypt $b_i = c_{i,2} \cdot c_{i,1}^{-x_i} = b_i \cdot y_i^{r_m} \cdot (g^{r_m})^{-x_i}$.
  2: Compute $k_i = (Q_i)^{x_i b_i} = g^{q_i x_i b_i} \pmod p$.
  3: Decrypt $\mathsf{bal}_i = \mathsf{AES\text{-}Dec}_{k_i}(\mathsf{ebal}_i)$.
```

where $T_i = \mathsf{bal}_i \,\|\, \mathsf{vote}_i$. If $\mathsf{bal}_i \in \mathcal{B}'$, the tallying authority examines the attached vote sequence and increments the counter of each candidate accordingly. Each election may specify its own rules that disqualify votes which are cast incorrectly. For example, voting *yes* for two opposing candidates.

VI. Security and Privacy Analysis

In this section, a formal proof of security and privacy is presented.

A. Double-Voting

Unlike the conventional elections where eligible voters are granted the right to voice their opinion exactly one time, there are many possible new security and privacy issues for electronic and remote voting. The malicious act of attempting to cast more than one vote is referred to as *double-voting* and aims to give an election candidate an advantage in winning the election over others. In countries such as the United States, while the majority of states prohibit voting twice in the same election, only a few of them prohibit voting in more than one state [23]. This means that eligible voters could register in more than one state and attempt to double-vote. The process of detecting and penalizing such voters becomes expensive and challenging. In remote electronic-voting systems, voters receive digital ballots and cast their votes remotely rather than visiting a polling station. Since our proposed scheme uses the blockchain as a bulletin board to permanently store votes, it becomes simple to identify double-voting attempts that reuse digital ballots. However, the reuse of digital ballots is not the only method to attempt double-voting. Adversaries may attempt to double-vote by obtaining undeserved voting credentials giving them the right to cast more votes. We formally prove that an election running d-BAME is secure against such attempts.

Definition 1 (Secure against double-voting)**.** *A voting scheme is said to be secure against double-voting if no PPT adversary is able to forge a digital ballot that is digitally signed by the registrar.*

Theorem 1. *It is infeasible for any adversary to generate a legitimate ballot that can be used to cast a vote correctly if the DDH assumption holds.*

Proof. Each ballot $\mathsf{bal}_i \in \mathcal{B}'$ is digitally signed by the registrar before being distributed among the anonymous voters. For the adversary to generate legitimate ballots, it must be able to forge a signature of a ballot $\mathsf{bal}'_i = t_i \parallel \mathsf{EG\text{-}Sign}(t'_i)$. This requires the adversary to learn the secret key x_r of the registrar or find collisions such that $h(\mathsf{bal}'_i) = h(\mathsf{bal}_i)$. Both operations can be reduced to the discrete logarithm problem. Therefore, it is infeasible for the adversary to generate a signed ballot correctly and the proof is complete. □

B. Voter Anonymity

To preserve the anonymity of voters, an adversary should not be able to link any vote to a specific voter. The proposed scheme relies on a secure multi-party computation performed by parties of different allegiances to address this issue. As discussed in Section V-C, it requires a minimum of two conflicting parties to participate during the ballot distribution process. As demonstrated, a moderator conceals the public key y_i of the voter using a blind factor f_i and associates it with a random value selected from the permutation $\sigma(i)$. On the other hand, the registrar selects a ballot randomly and assigns it to the anonymous voter. The moderator and registrar would need to collude for ballots to be linked to the identities of voters. We assume collusion is not in the best interest of any of these parties, therefore, voter anonymity can be preserved. Our proposed scheme is considered to be secure under this assumption if (i) the probability of the registrar to identify a public key y_i that has been randomly selected from a two-element public key chosen by the adversary and blinded does not significantly exceed $\frac{1}{2}$ and (ii) the moderator cannot derive an encrypted ballot. Given these stringent conditions, any other adversary should not be able to break voter anonymity since it is assumed that access to voter identities and/or ballots is inadequate in comparison to both the registrar and moderator.

First, using the Indistinguishability under Chosen-Plaintext Attack (IND-CPA) Security Game, we provide a formal security proof that the registrar cannot derive a blinded public key. We denote the DDH oracle as $\mathcal{O}_1 = (\mathsf{BFGen}, \mathsf{Blind}, \mathsf{Rec})$, where BFGen is the blind factor generation function, Blind is the blind function, and Rec is the recovery function. The IND-CPA game consists of a set of interactions between two PPT machines, an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ acting as the moderator.

1) $\mathcal{C}$ computes a blind factor $f = \mathsf{BFGen}(1^k)$ and keeps it secret.
2) Since $\mathcal{A}$ does not have access to $\mathcal{O}_1$, it may request that $\mathcal{C}$ blinds for it as many public addresses as it likes during any time of the game. $\mathcal{A}$ then computes two public addresses y_0 and y_1 to be challenged against and sends them to $\mathcal{C}$.
3) $\mathcal{C}$ uniformly and randomly selects $\mu \in_r \{0, 1\}$ then computes $y' = \mathsf{Blind}(f, s, y_\mu)$, where s represents a randomness state to diversify the blind process and is a value that has not been used in any of the previously computed ciphertexts. Next, $\mathcal{C}$ sends y' to $\mathcal{A}$.
4) $\mathcal{A}$ outputs a guess μ' of μ. $\mathcal{A}$ wins the security game if $\mu' = \mu$ and loses otherwise.

An adversary that can derive which public key was blinded in polynomial time may be able to identify the identities of the voters and link them to the ballots being assigned. The DDH assumption implies that the adversary is unable to get a non-negligible advantage from the IND-CPA Security Game in determining the public key that is blinded.

Second, we prove that the moderator cannot derive an encrypted ballot by the registrar.

Definition 2 (Voter anonymity)**.** *A voting scheme is said to preserve the anonymity of voters if no PPT adversary is able to get a non-negligible advantage in deriving the identity of the voter of any cast ballot.*

Theorem 2. *The proposed scheme preserves the voter anonymity if the DDH assumption holds.*

Proof. We first begin by proving that the registrar cannot derive the identity of a voter it has assigned a ballot to. Assume there is an adversary that has non-negligible advantage ε, then $\mathsf{Adv}_\mathcal{A} > \frac{1}{2} + \varepsilon$. We construct a simulator $\mathcal{A}$ that can distinguish a DDH element from a random element with advantage ε. Let $\mathbb{G}$ be a publicly chosen multiplicative cyclic group of prime order p. The DDH challenger begins by selecting the random parameters: $a, b \in_r \mathbb{Z}_p^*$. Let $g \in \mathbb{G}$ be a generator and Y is defined as $Y = g^{ab} \pmod{p}$ if $\mu = 0$, and $Y = g^c \pmod{p}$ for some random $c \in_r \mathbb{Z}_p^*$ otherwise, where $\mu \in_r \{0, 1\}$. The simulator acts as the challenger in the IND-CPA game.

1) $\mathcal{C}$ chooses a blind factor $f \in_r \mathbb{Z}_p^*$ and state $s^* \in_r \mathbb{Z}_p^*$ then computes $s = s^* + ab$ and keeps them secret.
2) $\mathcal{A}$ chooses two secret keys $x_0, x_1 \in_r \mathbb{Z}_p^*$ then computes their corresponding public addresses y_0 and y_1 and sends them to $\mathcal{C}$.
3) $\mathcal{C}$ uniformly and randomly selects $\mu \in_r \{0, 1\}$ then computes $y^* = \mathsf{Blind}(f, s, y_\mu) = y_\mu^f g^s = g^{x_\mu f} g^{s^* + ab} =$

$g^{x_\mu f} g^{s^*} Y \pmod p$, where $Y = g^{ab} \pmod p$. Next, $\mathcal{C}$ sends y^* to $\mathcal{A}$.

4) $\mathcal{A}$ outputs a guess μ' of μ. $\mathcal{A}$ wins the security game if $\mu' = \mu$ and loses otherwise.

Given $\mathcal{A}$, if $Y = g^{ab} \pmod p$, then y^* is a valid ciphertext, $Adv = \varepsilon$ and

$$\mathsf{Pr}\left[\mathcal{A}\left(g, g^a, g^b, Y = g^{ab}\right) = 1\right] = \frac{1}{2} + \varepsilon. \tag{16}$$

If $Y = g^c \pmod p$ *or* $Y \neq g^{ab} \pmod p$ then y^* is nothing more than a random value to the adversary. Therefore,

$$\mathsf{Pr}\left[\mathcal{A}\left(g, g^a, g^b, Y = g^c\right) = 1\right] = \frac{1}{2}. \tag{17}$$

From equation (16) and equation (17), we can conclude that

$$\left|\mathsf{Pr}\left[\mathcal{A}\left(g,g^a,g^b,Y=g^{ab}\right)=1\right]-\mathsf{Pr}\left[\mathcal{A}\left(g,g^a,g^b,Y=g^c\right)=1\right]\right|=\varepsilon.$$

The simulator plays the DDH game with a non-negligible advantage which contradicts the DDH assumption. Therefore, neither the registrar nor any other adversary can get any advantage ε to derive the identity of a voter that has been assigned a known ballot.

Concurrently, the moderator has no advantage in deriving the assigned ballot it transmits to the voter during the ballot transmission process. Each bal_i is encrypted by the registrar as shown in equation (7) prior to being shared with the moderator. The encryption key k_i is generated by the registrar based on the blinded public key $y_i^{`}$ and a value $q_i \in_r \mathbb{Z}_p^*$ selected randomly. This derivation can be reduced to the discrete logarithm problem making it infeasible for the moderator or any other adversary to derive k_i and decrypt ebal_i.

Therefore, our proposed scheme preserves voter anonymity against the registrar, moderator, and any other adversary, and the proof is complete. □

C. Coercion-Resistance and Ballot Unlinkability

In the proposed scheme, casting a vote traces back to the voters encrypting their desired votes as shown in equation (13). To prove that the proposed scheme is coercion-resistant, we use the IND-CPA game consisting of a DDH oracle $\mathcal{O}_2 = (\mathsf{KeyGen}, \mathsf{D\text{-}Enc}, \mathsf{D\text{-}Dec})$, where KeyGen is a public key pair generator, $\mathsf{D\text{-}Enc}$ and $\mathsf{D\text{-}Dec}$) are the encryption and decryption functions shown in equations (13) and (15) respectively. The IND-CPA game is a set of interactions between two PPT machines, an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1) $\mathcal{C}$ computes two pairs of keys $\mathsf{KeyGen}(1^k) \rightarrow (y_0, x_0)$ and $\mathsf{KeyGen}(1^k) \rightarrow (y_1, x_1)$ then sends the public keys y_0 and y_1 to $\mathcal{A}$ while keeping x_0 and x_1 secret.
2) $\mathcal{A}$ has access to $\mathcal{O}_2$ and can encrypt as many $T = \mathsf{bal} \,\|\, \mathsf{vote}$ of its choice. Next, $\mathcal{A}$ chooses a $T_0 = \mathsf{bal} \,\|\, \mathsf{vote}_0$ and $T_1 = \mathsf{bal} \,\|\, \mathsf{vote}_1$ then sends them to $\mathcal{C}$.
3) $\mathcal{C}$ uniformly and randomly selects $\mu \in_r \{0,1\}$ then computes $c^* = \mathsf{D\text{-}Enc}(g^v,\, T_\mu \cdot (y_r \cdot y_m)^v)$. Next, $\mathcal{C}$ sends c^* to $\mathcal{A}$.
4) $\mathcal{A}$ outputs a guess μ' of μ. $\mathcal{A}$ wins the security game if $\mu' = \mu$ and loses otherwise.

An adversary that can derive which T was encrypted efficiently may potentially be able to learn if voters have resubmitted their votes under the same ballots. In our proposed scheme, only the last cast vote with a legitimate ballot is counted towards the election. As a result, the adversary may be able to discover whether the coerced voter has behaved as instructed. The DDH assumption implies that the adversary is unable to get a non-negligible advantage from the IND-CPA Security Game in determining the T that was encrypted.

Definition 3 (Coercion-resistance)**.** *A voting scheme is said to be coercion-resistant if no PPT adversary is able to get a non-negligible advantage by performing the IND-CPA Security Game, i.e.* $\mathsf{Adv}_{\mathcal{A}} = Pr[\mu' = \mu_i] < \frac{1}{2} + \varepsilon$ *for any negligible* ε.

Theorem 3. *The proposed scheme is coercion-resistant if the DDH assumption holds.*

Proof. Assume there is an adversary that has non-negligible advantage ε, i.e., $\mathsf{Adv}_{\mathcal{A}} > \frac{1}{2} + \varepsilon$. We construct a simulator $\mathcal{A}$ that can distinguish a DDH element from a random element with ε. Let $\mathbb{G}$ be a publicly chosen multiplicative cyclic group of prime order p. The DDH challenger begins by selecting the random parameters: $a, b \in_r \mathbb{Z}_p^*$. Let $g \in \mathbb{G}$ be a generator and Y is defined as $Y = g^{2ab} \pmod p$ if $\mu = 0$, and $Y = g^c \pmod p$ for some random $c \in_r \mathbb{Z}_p^*$ otherwise, where $\mu \in_r \{0,1\}$. The simulator acts as the challenger in the following game:

1) $\mathcal{C}$ chooses the parameters $x_0^*, x_1^* \in_r \mathbb{Z}_p^*$ then computes $x_0 = x_0^* + a$ and $x_1 = x_1^* + a$. Next, it simulates $y_0 = g^{x_0} \leftarrow g^{x_0^*+a} \pmod p$ and $y_1 = g^{x_1} \leftarrow g^{x_1^*+a} \pmod p$. Finally, it sends y_0 and y_1 to $\mathcal{A}$ and keeps x_0 and x_1 secret.
2) $\mathcal{A}$ chooses chooses a $T_0 = \mathsf{bal} \,\|\, \mathsf{vote}_0$ and $T_1 = \mathsf{bal} \,\|\, \mathsf{vote}_1$ then sends them to the simulator.
3) $\mathcal{C}$ uniformly and randomly selects $\mu \in_r \{0,1\}$ then computes $c^* = \mathsf{D\text{-}Enc}_{y_0,y_1}(T_\mu) = (g^b, T_\mu \cdot (g^{(x_o^*+a)b} \cdot g^{(x_1^*+a)b})) = (g^b, T_\mu \cdot (g^{x_0^*} \cdot g^{x_1^*} \cdot g^{2ab})) = (g^b, T_\mu \cdot (g^{(x_o^*+a)b} \cdot g^{(x_1^*+a)b})) = (g^b, T_\mu \cdot Y \cdot (g^{x_0^*} \cdot g^{x_1^*}))$, where $Y = g^{2ab} \pmod p$. Next, $\mathcal{C}$ sends c^* to $\mathcal{A}$.
4) If $\mathcal{A}$ guesses the correct value, the challenger outputs 0 to indicate that $Y = g^{2ab}$, or 1 to indicate that $Y = R$, a random group element in $\mathbb{G}$.

Given $\mathcal{A}$, if $Y = g^{2ab} \pmod p$, then c^* is a valid ciphertext, $Adv = \varepsilon$ and

$$\mathsf{Pr}\left[\mathcal{A}\left(g, g^a, g^b, Y = g^{2ab}\right) = 1\right] = \frac{1}{2} + \varepsilon. \tag{18}$$

$Y = g^c \pmod p$ *or* $Y \neq block^{2ab} \pmod p$ then y^* then c^* is nothing more than a random value to the adversary. Therefore,

$$\mathsf{Pr}\left[\mathcal{A}\left(g, g^a, g^b, Y = g^c\right) = 1\right] = \frac{1}{2}. \tag{19}$$

From equation (18) and equation (19), we can conclude that

$$\left|\mathsf{Pr}\left[\mathcal{A}\left(g,g^a,g^b,Y=g^{2ab}\right)=1\right]-\mathsf{Pr}\left[\mathcal{A}\left(g,g^a,g^b,Y=g^c\right)=1\right]\right|=\varepsilon.$$

The simulator plays the DDH game with a non-negligible advantage which contradicts the DDH assumption. Therefore the adversary cannot have an advantage ε and the proof is complete. □

Corollary 1 (Ballot unlinkability)**.** *It is computationally infeasible to link any vote to its voter, or link multiple votes cast by one voter (including the ones created for coercion).*

Proof. In Theorem 2 we prove that d-BAME preserves voter anonymity. Correspondingly, it is computationally infeasible for an adversary to link a ballot to the voter. In Theorem 3 we also show that d-BAME is coercion-resistant during the casting votes phase. This allows voters to cast their votes multiple times in an election having only the predetermined vote as described later in Section IX-C counted toward the election results. By design and to provide voter verifiability once the election results are disclosed, the coercer may learn that the voter has not voted as instructed. □

D. Election Results Manipulation

The proposed scheme utilizes a blockchain as its public bulletin board for voters to cast their votes. It is secure against election result manipulation if it is run on a blockchain that adopts the longest chain rule and the computational power of the adversary in the blockchain network is no more than 50%. Voters interact with a voting smart contract that accepts a vote in the form shown by equation (13) as input. At the end of the voting phase, the tallying authorities scan the blockchain logs to collect all votes that have been cast during the voting phase. Votes that pass the validation are counted towards the election results while those that fail are discarded. Valid votes are posted to the blockchain along with their corresponding ballots to announce election results and allow voter validation. Voters can individually recognize that their votes have been counted correctly towards the final election results.

Before casting their votes to the blockchain, voters are required to encrypt their votes under public keys of both the registrar and moderator. This encryption acts as a secure envelope as it conceals the actual vote during the voting phase. It ensures that all encrypted votes can only be opened through decryption when both parties disclose their private keys. Ballots associated with votes are checked against the published $\mathcal{B}'$ in the blockchain. Votes with legitimate ballots are accepted or rejected otherwise, making this process similar to the signature verification on mail-in ballots in the US general election.

VII. Performance Evaluation

In this section, we present a performance evaluation for d-BAME and compare it with two blockchain-based schemes presented in [14] and [15]. In comparison to our work, both schemes perform some cryptographic operations in their deployed election smart contracts. Therefore, d-BAME can achieve a significant performance advantage and is more suitable for mobile e-voting.

A. Computational Costs

We formulate the computational costs of each voting stage for the three schemes based on the number of multiplication (M) and exponentiation (E) operations. Table II summarizes the number of these two operations for each scheme.

1) Setup: In [14], the setup consists of an admin that authenticates voters with their Ethereum user-controlled accounts then updates a whitelist of voters to include all eligible voters. However, the paper did not discuss any cryptographic operations in the authentication process. We assume that the setup for this scheme does not require any M or E operations.

On the contrary, [15] requires cryptographic primitives like Paillier encryptions, proof-of-knowledge (PoK), and Short Linkable Ring Signatures (SLRSs) computed over the blockchain platform.

In comparison to both schemes, d-BAME requires the moderator and registrar to generate their public key pairs, y_m and y_r, that are used during the entire election process, imposing a single E operation for each.

2) Voter Registration: At the voter registration stage, [14] requires each voter to generate a key pair at a cost of one E operation. Voters must also generate a non-interactive Zero Knowledge Proof $\mathsf{ZKP}(x_i)$ to prove knowledge of their secret key sk_i. Specifically, it uses a Schnorr proof [24] made non-interactive using the Fiat-Shamir heuristic [25], resulting in an additional M and E operation. Once derived, voters broadcast pk_i and $\mathsf{ZKP}(x_i)$ through the specified election smart contract.

In [15], voter registration requires interaction between the voters and an election smart contract. Initially, eligible voters obtain the SLRS parameters generated during the setup phase and generate their SLRS key pairs. Voters then send their public keys to the blockchain through the smart contract to verify their signature.

In comparison, d-BAME relies on a registrar to facilitate voter registration. Voters begin by generating their key pairs (sk_i, pk_i). After verifying the eligibility of voters, the registrar signs their public keys and adds them to the electoral roll. The total signatures for registering all voters are $2n\mathsf{M}$ and $n\mathsf{E}$ operation.

3) Acquiring Ballots: To acquire a ballot, [14] initially requires the election smart contract to verify all n received $\mathsf{ZKP}(x_i)$. A single verification costs one M and 2E operations. Next, the smart contract generates all n digital ballots, where a single ballot generation requires a total of $n\mathsf{M}$ operations. Voters acquire their corresponding generated ballots and use them in the next stage during vote casting.

In [15], voters are not required to perform any computations to acquire a digital ballot. Instead, they can immediately cast their votes once registration is complete.

In comparison to both schemes, d-BAME involves voter engagement with the moderator and registrar. Initially, the registrar generates the set of random and digitally signed ballots. Next, the voters interact with the moderator, who obscures their identities from the registrar while facilitating ballot distribution. The total operations performed by a voter, the moderator, and the registrar are $2\mathsf{M}+2\mathsf{E}$, $2n\mathsf{M}+6n\mathsf{E}$, and $2n\mathsf{E}$ operations, respectively.

4) Casting Vote: To cast a vote, [14] requires each voter to perform one M and 2E operations. Each voter must also generate another $\mathsf{ZKP}(x_i)$ to prove that they have voted either

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3074877, IEEE Internet of Things Journal

TABLE II: Performance Comparison.

	Setup						**Voter Reg.**						**Acquiring Ballots**						**Casting Votes**						**Tabulation**					
Scheme	[14]		[15]		d-BAME		[14]		[15]		d-BAME		[14]		[15]		d-BAME		[14]		[15]		d-BAME		[14]		[15]		d-BAME	
Operation	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E	M	E
Voter	0	0	0	0	0	0	1	2	1	1	0	1	0	0	n/a		2	2	2	3	$2(m+b+1)$	$3m+b+9$	2	2	0	0	0	0	0	0
Moderator	n/a		2	0	0	1	n/a		0	0	0	0	n/a		n/a		$2n$	$6n$	n/a		0	0	0	0	n/a		10	5	0	0
Registrar	n/a		5+κ	1+2κ	0	1	n/a		0	0	$2n$	n	n/a		n/a		0	2n	n/a		0	0	0	0	n/a		n/a		0	0
Tallying Authority	n/a		n/a		0	0	n/a		n/a		0	0	n/a		n/a		0	0	n/a		n/a		0	0	n/a		n/a		$2n$	$2n$
Smart Contract	0	0	0	0	0	0	0	0	0	n	0	0	$2n$	$2n$	n/a		0	0	0	0	$n(1+b)$	$5n$	0	0	n	0	1	$3+m$	0	0

yes or no. The generated $\mathsf{ZKP}(x_i)$ requires an additional M and E operation.

In [15], voters can vote for multiple candidates. The selected candidates are encrypted using Paillier homomorphic encryption, which is about 4 times slower than the ElGamal encryption since its operational parameters are twice of the size of the ElGamal scheme. The voters then compute a PoK to prove that they correctly encrypt a candidate and send their encryption and derived PoKs to the smart contract for verification. If valid, the smart contract signs the result and returns it to the voter. Upon receiving it, voters validate the signature and generate SLRS over the result to finalize their votes and cast it to the blockchain. Next, they generate a PoK for their signatures and send it to the smart contract for verification.

In comparison, d-BAME requires a voter to double encrypt bal_i using ElGamal encryption and attach it to the vote_i where, vote_i is a sequence of bits representing each candidate, 1 if voting for cand_k and 0 if not voting for cand_k. The total operations performed by a voter include $2\mathsf{M}$ and $2\mathsf{E}$ operations.

5) Tabulation: In [14], tabulation requires verifying each $\mathsf{ZKP}(x_i)$ submitted the voters, imposing one M and $2\mathsf{E}$ operations per verification. Next, the tally is computed requiring $n\mathsf{M}$ operations.

In [15], the election smart contract first adds all published and encrypted votes then signs the result and sends it to the admin. The admin verifies the signature and decrypts the sum using homomorphic decryption. The admin must also decrypt and verify the correctness of the PoK and sends the results back to the smart contract. The smart contract finally verifies the correctness of the information sent.

In comparison to both schemes, for d-BAME, the tallying authority is required to double decrypt each existing vote using the revealed moderator and registrar private keys, x_m and x_r. This imposes a computational cost of $2n\mathsf{M}$ and $2n\mathsf{E}$ operations.

B. Scalability

Ethereum 1.0 blockchain is based on the Proof-of-Work (PoW) consensus mechanism, limited to approximately 15 transactions per seconds. More recently, Ethereum 2.0 has launched utilizing the Proof-of-Stake (PoS) consensus mechanism, promising up to 100,000 transactions per second in two years [26]. Running an election for the approximately 150 million Americans that voted in the 2020 U.S. presidential election utilizing d-BAME over the Ethereum 2.0 will suffice to complete the casting votes phase in approximately 25 minutes. Our recommendation to utilize Ethereum blockchain is solely based on its wide popularity and adoption. It may not guarantee the best performance or cost efficiency. In fact as the blockchain research continues to grow, other blockchains may emerge to be more scalable and cost efficient.

VIII. Empirical Results

In this section, we present our empirical results of various simulations of d-BAME. We implement two versions of d-BAME, a desktop application, and a smartphone application. The desktop application is implemented using Maple v16 and we simulate it on a MacBook Air running OS X 10.13.6 equipped with 2 cores, 1.8 GHz Intel Core i5, and 8 GB 1600 MHz DDR3. The smartphone application is developed over Xcode version 11.2.1 and is written in Swift 5. We use the BigInt library[1] to perform large number cryptographic computations. Our smartphone simulations are performed over an iPhone XR running iOS 13.2.2 equipped with the Apple A12 Bionic, a 64-bit ARMv8.3-A six-core CPU, with two cores running at 2.49 GHz.

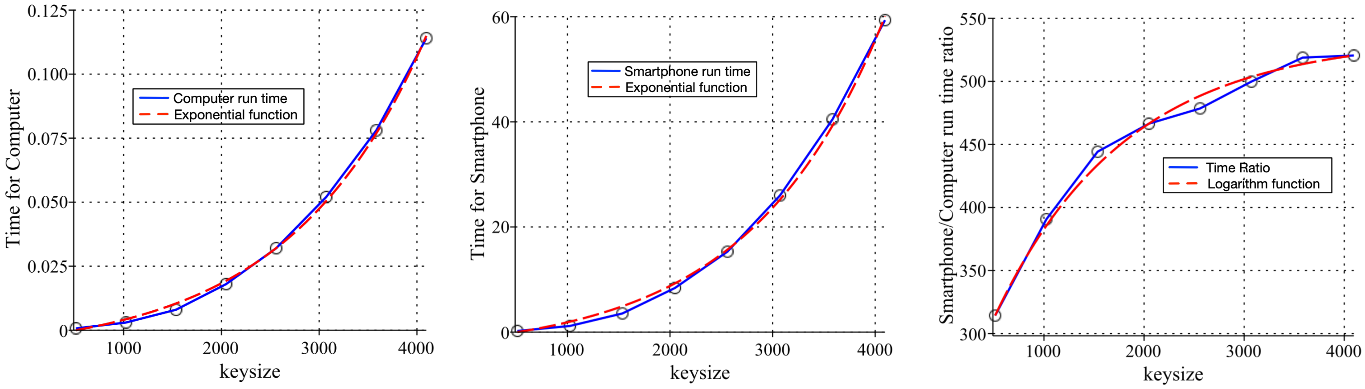
Both applications interact with our Solidity-based smart contract that we deploy over the Ethereum Ropsten testnet blockchain to cast the encrypted votes at the end of the voting stage. The main inputs to the smart contract are the two encrypted vote components, $B_i = (c_{i,3},\, c_{i,4})$, as depicted in equation (13). For our desktop application, we utilize MetaMask[2], a browser plug-in that allows voters to manage their Ethereum wallets and call our deployed smart contract to cast their votes. For our smartphone application, we incorporate the web3swift library[3] into our code to provide the voters with the same functionality to cast their votes using their mobile devices. We note that the performance relies on the selection of software packages. Our selection does not guarantee the best performance. Instead, it shows that d-BAME is suitable for large-scale elections.

For an election, d-BAME is run by each voter on either a desktop or mobile device. After vote casting is complete, tabulating the votes is a job performed by the tallying authority, which is generally performed offline using powerful computers. Therefore, we assume that all stages are performed over a mobile device while tabulation is performed by anyone that has access to a desktop machine. In comparison to other schemes, d-BAME is the only scheme that allows voters to generate the election results themselves if they wish and can afford the required computational requirements.

[1] https://github.com/attaswift/BigInt

[2] https://github.com/MetaMask/metamask-extension

[3] https://github.com/matter-labs/web3swift



(a) (b) (c)

Fig. 3: Time comparison for various key sizes: (a) in computer, (b) in smartphone, (c) smartphone/computer.

Therefore, we focus our analysis on investigating the different time costs for casting a vote for both a computer and a smartphone. Specifically, we measure the time costs to perform the double encryption computation presented in equation (13). Fig. 3 shows the time costs we obtain under eight different encryption key sizes. The presented time costs are the average time costs of ten different trials for each key size.

Based on our results, we come to various conclusions. Fig. 3(a) shows that with maximized security at a key size of 4096 bits, voters can cast their votes in approximately 0.11 seconds using a desktop. Correspondingly, Fig. 3(b) shows that to maintain the same level of security while running d-BAME over a smartphone, it would take less than a minute to cast a vote. In both cases, the time increases exponentially with the increasing of key sizes. Today, in practice, key sizes of 2048 bits are generally considered secure, hence, casting a vote through a mobile device may even be reduced to approximately 8.36 seconds. The difference between the time costs and of running d-BAME over a desktop and a smartphone is evident because of the processor capabilities we describe above. Desktop machines are generally built with more powerful processors giving them a conspicuous advantage over smartphones. However, the purpose of this analysis is to prove that even with this advantage, it is still feasible to run d-BAME over a smartphone and achieve acceptable results.

To further analyze our findings, we also measure the smartphone to desktop time cost ratio and observe its change as we increase the key size. Fig. 3(c) presents these results showing that as the key size increases, the ratio increases logarithmically. This suggests that, beyond a certain key size, the advantage of running d-BAME over a desktop versus running it over a smartphone decays as the keys grow in size. For example, as shown in Fig. 3(c), the smartphone/desktop ratio at the key size of 2048 bits is 466 and increases to 478 with the key size of 2560 bits, showing a 2.5% increase. This increase is smaller when compared to the ratio increase between the key size of 1536 bits, where the ratio is 444, and that at the key size of 2048 bits, showing an increase of 5%. This pattern can be observed between all key sizes shown in the figure.

IX. Design Trade-offs

In this section, various implementation trade-offs are discussed. These trade-offs enable elections to adjust to different d-BAME implementations based on the required design goals.

A. Universal Verifiability vs. Coercion-Resistance

In traditional paper ballot-based voting systems, voters cast their votes in a physically isolated environment without facing interference from coercers throughout the election process. However, universal verifiability is limited to monitoring ballots as they are counted which is prone to significant error. Even when utilizing on-site computerized voting machines to provide features such as voter and universal verifiability, voters must trust that these machines are secure, privacy-preserving, and software independent, which means that an undetected change or error in its software cannot cause an undetectable change or error in an election outcome [27].

On the contrary, in mobile e-voting systems, voting is performed remotely by the voters via their personal devices such as desktops or mobile devices, allowing features such as voter and universal verifiability features to be more easily implemented. Assuming the devices are secure, voters need just to trust that the installed voting software is legitimate. However, with mobile e-voting, the coercers may be able to interfere with any phase of the voting process. To address this concern, a stronger form of private voting known as coercion-resistance emerged [6]. A coercion-resistant voting system is one that accounts for coercers that can engage with voters while they cast their votes remotely during an election. Engagement may be in the form of a coercer forcing voters to cast their votes in a specific form or even forcing them to divulge their voting credentials by easily blackmailing the voters. Other engagements may even include coercers willing to peacefully buy votes from the voters.

In Section V, we showed how two conflicting parties facilitate the anonymous distribution of our uniquely generated ballots. Under this assumption, it is not in any interest of either party to share any information during ballot distribution, thus, the identities of voters and their anonymously assigned ballots remain concealed. Consequently, voter verifiability is achieved since voters can recognize their ballots. While the uniqueness of the ballots ensures voter verifiability, it also gives rise to the coercion problem. Coercers that can get access to the ballots of voters will also be able to learn if these voters have behaved as instructed once the election results are finalized and disclosed, i.e. weak coercion-resistance. This results in a clear conflict between the universal verifiability and coercion-resistance features. Our proposed work is designed to achieve a trade-off between these two characteristics, allowing an election to favor one over the other.

a) Voter and Universal Verifiability: In the case that universal verifiability is preferred, at the end of the tabulation phase, the registrar and moderator both disclose their private keys and the tallying authority publishes the results of the election on the blockchain, which allows all voters to verify that their votes have been counted properly toward the election. Alternatively, votes can be published along with their corresponding ballots over the blockchain as a proof of the legitimacy of the election results. Any individual can validate that all votes have been cast and counted properly. This design also prevents the registrar from attempting to issue unassigned ballots to voters in favor of a particular candidate. In this setting, the proposed scheme provides weak coercion-resistance. Once the election results are disclosed, the coercers would be able to find out whether the coerced votes are being counted toward the election.

b) Strong Coercion-Resistance: A practical coercion-resistant voting system should be *receipt-free*, allowing voters to evade proving to their coercers how they voted. This property requires a voting system to be designed in a way so that it does not generate any legitimate evidence that may leak information about the votes. By design, our proposed scheme requires voters to encrypt their votes as shown in Equation (13). This can be viewed as a receipt and may allow information to be leaked on how voters have voted. Therefore, if strong coercion-resistance is required, ballot verification can be delegated to just the registrar and the moderator. The registrar and the moderator jointly decrypt and publish the results excluding the ballots. Given that the registrar and the moderator are unlikely to collude and that their verifications must match, voters can trust that the election result integrity is maintained. However, this limits voter and universal verifiability.

B. Receipt-Freeness

Our proposed scheme may be modified to offer receipt-freeness at an additional computational cost. Once a vote is cast as described in equation (13), the moderator can decrypt then re-encrypt it as follows:

$$B_i' = (c_{i,3} g^u, c_{i,4} c_{i,3}^{-x_m} y_r^u) = (g^{v+u}, T_i y_r^{v+u}) = (c_{i,3}', c_{i,4}'), \quad (20)$$

where $u \in_r \mathbb{Z}_p^*$. This method conceals the ballots of voters preventing them from identifying them during the vote casting phase. Simultaneously, once tallying is performed and the election results are disclosed, voter and universal verifiability can still be performed since ballots are unconcealed.

C. Counting First Ballot vs. n^{th} Ballot

In our proposed scheme, voters receive unique digital ballots which they use to cast their votes to the blockchain. This design grants voters the ability to cast their votes multiple times while identifying those trying to double vote. As a result, only one cast vote corresponding to each legitimate voter is counted toward the final election results. This feature allows elections to specify different policies on which vote to be counted based on the election needs/circumstances. Therefore it becomes a trade-off between selecting the first ballot vs. n^{th} ballot. Elections favoring counting the first vote may be more prone to coercion. It becomes more difficult for coercers to monitor and detect how voters have voted since they must be physically present in a timely manner among the voters to force them to give up their credentials. In large-scale elections, this becomes infeasible due to the geographical distribution of voters. Thus, the overall effect of coercing voters to manipulate the election results becomes insignificant. On the other hand, other elections may favor counting the n^{th} cast ballot. This method may be advantageous to elections that prefer allowing their voters to change their minds and re-cast their votes. However, with this extended time, coercers may have more time approaching a larger number of voters. In fact, elections that favor counting the last cast vote may be problematic in large-scale elections. Coercers that are able to obtain voters' credentials may wait until the last minute to cast all votes and manipulate the election results. Therefore, choosing this method may be more reasonable for small-scale elections with less possibility of coercion.

X. Conclusion

In this paper, we proposed d-BAME, a novel blockchain-based and remote electronic voting scheme. The proposed scheme is designed to run large-scale elections and aims at improving voter turnout. Our security and privacy analyses show that d-BAME is secure, preserves voter privacy, protects voters against coercers, and maintains the integrity of election results. In our performance analysis, we compare the computational complexity of d-BAME to two schemes and show that d-BAME is more appropriate for large-scale elections. Finally, in our empirical results, we present the results of running various simulations for d-BAME over both a desktop machine and a smartphone. Our results show that it is feasible to run d-BAME over a mobile device, hence improving the accessibility of elections.

References

[1] M. Bernhard, J. Benaloh, J. A. Halderman, R. L. Rivest, P. Y. Ryan, P. B. Stark, V. Teague, P. L. Vora, and D. S. Wallach, “Public evidence from secret ballots,” in *International Joint Conference on Electronic Voting*, pp. 84–109, Springer, 2017.

[2] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, “Analysis of an electronic voting system,” in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 27–40, IEEE, 2004.

[3] R. Gonggrijp and W.-J. Hengeveld, “Studying the nedap/groenendaal es3b voting computer: A computer security perspective,” in *Proceedings of the USENIX workshop on accurate electronic voting technology*, pp. 1–1, USENIX Association, 2007.

[4] P. Y. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, “Prêt à voter: a voter-verifiable voting system,” *IEEE transactions on information forensics and security*, vol. 4, no. 4, pp. 662–673, 2009.

[5] C. Culnane, P. Y. Ryan, S. Schneider, and V. Teague, “vvote: a verifiable voting system,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 18, no. 1, pp. 1–30, 2015.

[6] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 61–70, ACM, 2005.

[7] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 354–368, IEEE, 2008.

[8] B. Adida, "Helios: Web-based open-audit voting," in *USENIX security symposium*, vol. 17, pp. 335–348, 2008.

[9] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 393–403, Springer, 1995.

[10] D. Demirel, J. Van De Graaf, and R. S. dos Santos Araújo, "Improving helios with everlasting privacy towards the public," *EVT/WOTE*, vol. 12, 2012.

[11] J. Clark and U. Hengartner, "Selections: Internet voting with over-the-shoulder coercion-resistance," in *International Conference on Financial Cryptography and Data Security*, pp. 47–61, Springer, 2011.

[12] O. Spycher, R. Koenig, R. Haenni, and M. Schläpfer, "A new approach towards coercion-resistant remote e-voting in linear time," in *International Conference on Financial Cryptography and Data Security*, pp. 182–189, Springer, 2011.

[13] E. Zaghloul, T. Li, and J. Ren, "Anonymous and coercion-resistant distributed electronic voting," in *2020 International Conference on Computing, Networking and Communication*, IEEE, 2020.

[14] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Int. Conf. on Financial Cryptography and Data Security*, pp. 357–375, Springer, 2017.

[15] B. Yu, J. K. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, and M. H. Au, "Platform-independent secure blockchain-based voting system," in *International Conference on Information Security*, pp. 369–386, Springer, 2018.

[16] M. Schläpfer, R. Haenni, R. Koenig, and O. Spycher, "Efficient vote authorization in coercion-resistant internet voting," in *International Conference on E-Voting and Identity*, pp. 71–88, Springer, 2011.

[17] G. Tsoukalas, K. Papadimitriou, and P. Louridas, "From helios to zeus," *USENIX Journal of Election Technology and Systems (JETS)*, vol. 1, no. 1, pp. 1–17, 2013.

[18] A. Kiayias, T. Zacharias, and B. Zhang, "Demos-2: scalable E2E verifiable elections without random oracles," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 352–363, 2015.

[19] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1614–1625, 2016.

[20] F. Hao, P. Y. Ryan, and P. Zieliński, "Anonymous voting by two-round public discussion," *IET Information Security*, vol. 4, pp. 62–67, 2010.

[21] G. G. Dagher, P. B. Marella, M. Milojkovic, and J. Mohler, "Broncovote: Secure voting system using ethereum's blockchain," 2018.

[22] E. Zaghloul, T. Li, M. W. Mutka, and J. Ren, "Bitcoin and blockchain: Security and privacy," *IEEE Internet of Things Journal*, 2020.

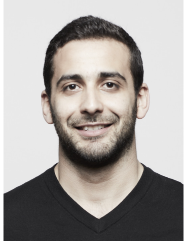
[23] National Conference of State Legislatures, "Double voting." http://www.ncsl.org/research/elections-and-campaigns/double-voting.aspx, 2018.

[24] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[25] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the Theory and Application of Cryptographic Techniques*, pp. 186–194, Springer, 1986.

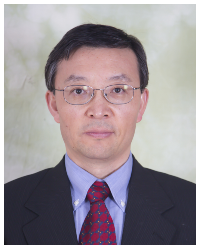
[26] "What is ethereum 2.0 and why does it matter?." https://decrypt.co/resources/what-is-ethereum-2-0, Dec 1, 2020.

[27] R. L. Rivest, "On the notion of 'software independence'in voting systems," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, pp. 3759–3767, 2008.



Ehab Zaghloul received his B.S. and M.Sc degrees in Computer Engineering from Arab Academy for Science and Technology (AAST), Alexandria, Egypt in 2012 and 2015 respectively. He received his Ph.D. degree in Electrical and Computer Engineering at Michigan State University (MSU), East Lansing, USA. His research interests include applied cryptography, secure and private cloud data sharing, electronic voting, cryptocurrencies, and blockchain.

Tongtong Li (SM'09) received her Ph.D. degree in Electrical Engineering in 2000 from Auburn University. From 2000 to 2002, she was with Bell Labs, and had been working on the design and implementation of 3G and 4G systems. Since 2002, she has been with Michigan State University, where she is now a Professor. Prof. Li's research interests fall into the areas of wireless and wired communications, wireless security, information theory and statistical signal processing, with applications in neuroscience. She is a recipient of a National Science Foundation (NSF) CAREER Award (2008) for her research on efficient and reliable wireless communications. Prof. Li served as an Associate Editor for IEEE Signal Processing Letters from 2007-2009, and an Editorial Board Member for EURASIP Journal Wireless Communications and Networking from 2004-2011. She served as an Associate Editor for IEEE Transactions on Signal Processing from 2012-2016.



Jian Ren (SM'09) received the BS and MS degrees both in mathematics from Shaanxi Normal University, and the Ph.D. degree in EE from Xidian University, China. He is an Associate Professor in the Department of ECE at Michigan State University. His current research interests include cybersecurity, cloud computing security, distributed data sharing and storage, decentralized data management, blockchain based e-voting and AI security and Internet of Things. He is a recipient of the US National Science Foundation (NSF) CAREER award in 2009. Dr. Ren served as the TPC Chair of IEEE ICNC'17, General Chair of ICNC'18 and Executive Chair of ICNC'19 and ICNC'20. Currently Dr. Ren serves as an Associate Editor for *IEEE Transactions on Mobile Computing*, *IEEE Internet of Things Journal*, *ACM Transactions on Sensor Networks (TOSN)* and a Senior Associate Editor for *IET Communications*. Dr. Ren is a senior member of the IEEE.